MÉTRICAS C&K APLICADAS AL DISEÑO ORIENTADO A ASPECTOS

Lorena Baigorria, Germán Montejano, Daniel Riesco
{flbaigor, gmonte, driesco}@unsl.edu.ar
Departamento de Informática, Universidad Nacional de San Luis
San Luis- CP 5700- Argentina

CACIC 2006- Ingeniería de Software

RESUMEN

La incesante búsqueda de mejorar el desarrollo de sistemas de software nos ha llevado a una nueva metodología; la Programación Orientación a Aspectos (POA), la cual ha surgido para mejorar la separación de las competencias en la programación de software, ésta se basa en tecnologías existentes, como la orientación a objetos (OO).
Debido a la necesidad de calcular los recursos necesarios para el desarrollo de sistemas de software se han realizado gran cantidad de investigaciones acerca de distintas métricas de software orientadas a objetos y procedurales, pero no para software orientado a aspectos.
Las métricas de software son formas de calificar los diseños de software. Se puede decir entonces que las métricas aplicadas a la POA son cruciales para determinar la efectividad de este paradigma como también de su uso en el diseño de sistemas de software.
Se propone en este trabajo distintas métricas aplicables a la POA sobre modelos desarrollados con UML (Unified Modeling Language) y las restricciones semánticas de las mismas se especificará con OCL (Object Constraint Language). Como así también la aplicación de las métricas C&K las cuales se aplican normalmente a la Orientación a Objetos y además analizar como influye la aplicación de la POA en ellas.

Keywords: Métricas, Orientación a Aspectos, UML, Perfil POA OCL, Métricas C&K, Métricas Orientadas a Aspectos.

1.INTRODUCCIÓN

Las métricas de software intentan medir la complejidad del software para predecir el costo total del proyecto y la evaluación de la calidad y la efectividad del diseño. Las métricas de software tienen múltiples aplicaciones en las distintas tareas de la ingeniería de software tales como pruebas, refactoring, gerenciamiento y mantenimiento [1].
La investigación de las métricas se va adaptando a las necesidades y nuevas propuestas de desarrollo de software, nuevos lenguajes y nuevos paradigmas de programación, como es el caso de la POA.
La *programación orientada a aspectos* (*POA*) es una nueva metodología de programación que aspira a soportar la separación de competencias para los aspectos tanto en el diseño como en la implementación de software. Los lenguajes orientados a aspectos se utilizan para lograr esta separación y así obtener las ventajas proporcionadas por esta metodología como: el fácil desarrollo y mantenimiento del software.
El software Orientado a Aspectos (OA)es básicamente diferente del software desarrollado usando métodos convencionales. Lo más cercano a éste es el software Orientado a Objetos; por esto las

métricas existentes para OO deben ajustarse a las características que distinguen el software OA de los demás.
El software desarrollado con esta metodología es más fácil de desarrollar y mantener, debido a la separación de competencias entre las componentes y los aspectos del software.
Al aplicar métricas al diseño de sistemas OA podemos obtener una visión objetiva de la calidad del diseño.
Como con cualquier métrica, el principal objetivo de las métricas OA se derivan del software convencional: mejor comprensión de la calidad del producto, estimación de la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto.

2.ORIENTACION A ASPECTOS

Los aspectos son propiedades de un software que tienden a atravesar sus funcionalidades principales. Consideramos un *aspecto a una unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación* [2].
La sincronización, el compartimiento de recursos y distribución son algunos ejemplos de aspectos. Normalmente los aspectos están entrelazados en el corazón de los componentes del sistema causando el problema de tener código desordenado.
La Orientación a Aspectos trata de modularizar este entrecruzamiento de competencias y encapsularlos en módulos en vez de tenerlos dispersos en los componentes del sistema.
Para lograr programas OA utilizamos los lenguajes OA. Los lenguajes orientados a aspectos definen una nueva unidad de programación de software para encapsular las funcionalidades que cruzan todo el código. Además, estos lenguajes deben soportar la separación de aspectos como la sincronización, la distribución, el manejo de errores, la optimización de memoria, la gestión de seguridad, la persistencia. De todas formas, estos conceptos no son totalmente independientes, y está claro que hay una relación entre los componentes y los aspectos, y que por lo tanto, el código de los componentes y de estas nuevas unidades de programación tienen que interactuar de alguna manera. Para que ambos (aspectos y componentes) se puedan mezclar, deben tener algunos puntos comunes, que son los que se conocen como *puntos de enlace,* y debe haber algún modo de mezclarlos.
El encargado de realizar este proceso de mezcla se conoce como *tejedor.* El tejedor se encarga de mezclar los diferentes mecanismos de abstracción y composición que aparecen en los lenguajes de aspectos y componentes ayudándose de los puntos de enlace [2].

3. UML (UNIFIED MODELING LANGUAGE)

El lenguaje unificado de modelado (UML) es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software, en particular un sistema Orientado a Objetos, el cual es un estándar de Object Management Group (OMG). Un artefacto es información usada para realizar un desarrollo de software.
La especificación de UML está definida por un metamodelo. Un metamodelo es un modelo que define el lenguaje para expresar otros modelos.
El lenguaje de modelado UML es el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa.
UML fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación (telecomunicaciones, comercio, sanidad, etc.) y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET o CORBA).

El hecho de que UML sea un lenguaje de propósito general proporciona una gran flexibilidad y expresividad a la hora de modelar sistemas. Sin embargo, hay numerosas ocasiones en las que es mejor contar con algún lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares. Esto sucede, por ejemplo, cuando la sintaxis o la semántica de UML no permiten expresar los conceptos específicos del dominio, o cuando se desea restringir y especializar los constructores propios de UML, que suelen ser demasiado genéricos y numerosos [3].

OMG define dos posibilidades a la hora de definir lenguajes específicos de dominio, y que se corresponden con las dos situaciones mencionadas antes: o bien se define un nuevo lenguaje (alternativa de UML), o bien se extiende el propio UML, especializando algunos de sus conceptos y restringiendo otros, pero respetando la semántica original de los elementos de UML (clases, asociaciones, atributos, operaciones, transiciones, etc.).
Hay situaciones en las que es suficiente con extender el lenguaje UML utilizando una serie de mecanismos recogidos en lo que se denominan Perfiles UML (UML Profiles); este es el caso de la POA. En la siguiente sección se mostrara como extender UML para modelar sistemas de software OA utilizando Perfiles UML.

3.1 EXTENSIÓN DE UML PARA LA POA

Los aspectos son intereses cruzados que tienden a atravesar los componentes principales del sistema, incrementando su interdependencia y reduciendo su reutilización. El desarrollo de software orientado a objetos provee soporte explicito para tratar con los intereses atravesados. Usando estas técnicas, los aspectos son ubicados en módulos separados que serán tejidos para formar una aplicación.
Usamos el término **modelado orientado a aspectos** para denotar el ámbito de los elementos de modelado para especificar los intereses cruzados en un alto nivel de abstracción. El modelado orientado a aspectos debería estar construido sobre un framework conceptual al que referiremos como el Modelo de Aspectos.
UML soporta el concepto de múltiples vistas. Esto permite a los diseñadores de software expresar varios requerimientos, decisiones de diseño e implementación usando cada una de las vistas independientes. Estas vistas están relacionadas unas a otras a través del metamodelo de UML, asegurando la coherencia del modelo de software. La extensibilidad de UML permite la adaptación para un ambiente de modelado específico [4].
Como se dijo anteriormente UML incluye un mecanismo de extensión en el propio lenguaje que permite definir lenguajes de modelado que son derivados de UML. De forma más precisa, el paquete Profiles de UML 2.0 [5] define una serie de mecanismos para extender y adaptar las metaclases de un metamodelo cualquiera (y no sólo el de UML) a las necesidades concretas de una plataforma (como puede ser .NET o J2EE) o de un dominio de aplicación (tiempo real, modelado de procesos de negocio, etc.).
En el caso de la POA se extenderá UML utilizando Perfiles; es decir no se modificará la semántica de UML, solo se particularizarán algunos conceptos.
Un Perfil se define en un paquete UML, estereotipado «profile», que extiende a un metamodelo o a otro Perfil. Tres son los mecanismos que se utilizan para definir Perfiles: estereotipos (*stereotypes*), restricciones (*constraints*), y valores etiquetados (*tagged values*). En la figura 1 se muestra el metamodelo que describe la POA.

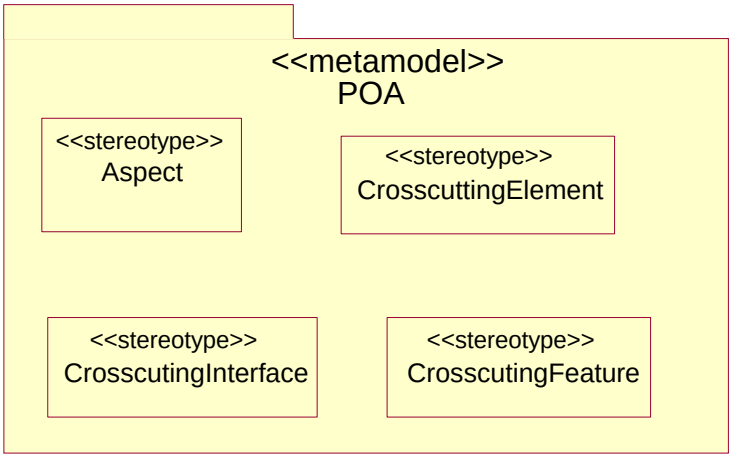

Fig.1 Metamodelo para la POA

Las entidades que forman la extensión del metamodelo son:

- **BaseElement** : Esta entidad representa a los elementos que podrían participar en una relación con aspectos, en nuestra extensión estos estarían representados por las clases.
- **CrosscuttingElement** : Esta entidad encapsula las entidades principales de nuestra extensión y esta compuesta por :
  - **Aspect** : Un aspecto representa la funcionalidad que atraviesa el sistema y es construido por un constructor derivado del elemento **Classifier** (Fig 1). Un aspecto comprende un conjunto de características y un conjunto de CrosscuttingInterface.
  - **CrosscuttingInterface :** Cada una de las CrosscuttingInterface ("Puntos de corte") describe un conjunto mínimo de propiedades requeridas para las clases, cuyo estado y comportamiento son extendidos de la estructura y comportamiento del aspecto al que están relacionados.
  - **CrosscuttingFeature :** Esta entidad modela elementos que describen una característica ("Puntos de Unión") que serán combinados con elementos BaseElement.

A partir de este metamodelo que se muestra en la figura 1, el Perfil UML que va ha representarlo va a estar escrito como un paquete UML estereotipado <<profile>>. En la figura 2 se nuestra el Perfil POA para la extensión de UML, de esta manera se pueden modelar sistemas Orientados a Aspectos con UML.

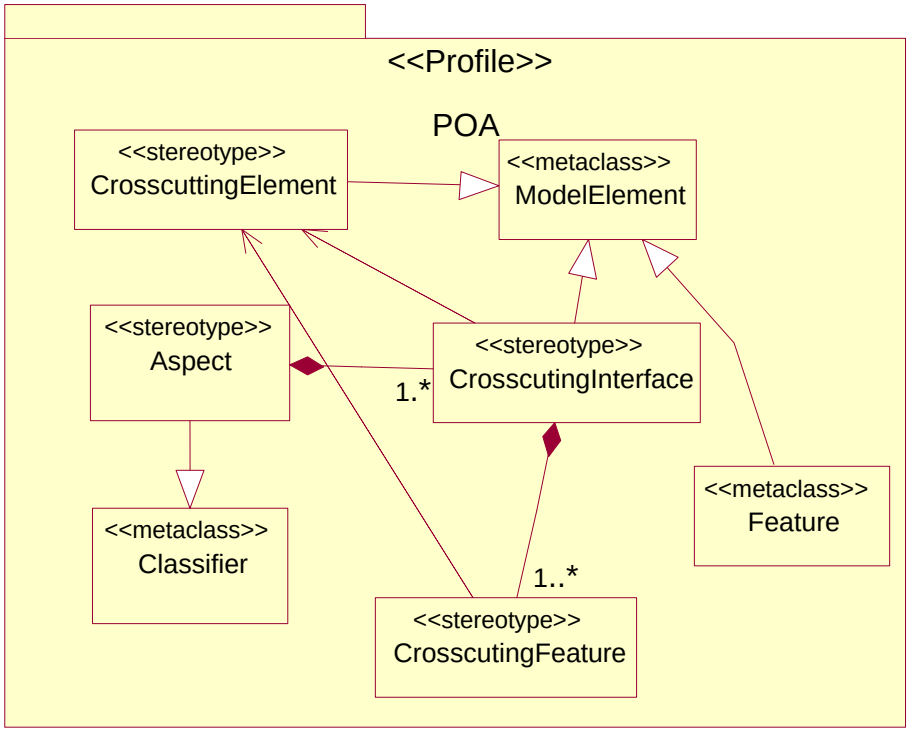

Fig. 2: Perfil POA

4. OCL (OBJECT CONSTRAINT LANGUAGE)

El Object Constraint Language es un lenguaje de expresión que permite describir las restricciones sobre modelos Orientados a Objetos (OO) y otros artefactos de modelado. Una restricción es una condición o regla que debe cumplirse sobre uno o más valores de un modelo de un sistema OO. OCL es parte de UML.
En la práctica, hay muchas restricciones sobre los objetos y las clases en un modelo OO que no pueden especificarse gráficamente. Podemos llamar a estas reglas restricciones. Una restricción da información extra sobre un modelo OO o un artefacto.
Las restricciones se aplican siempre a elementos de modelos o de sistemas OO, y siempre restringen valores de esos elementos. Las restricciones deben ser verdaderas en un momento dado para que el modelo o el artefacto sea validado. Se utiliza al metamodelo de OCL para definir restricciones propias de UML. OCL no es sólo usado para especificar restricciones también se utiliza para definir restricciones a los metamodelos definidos por la OMG [6].
Debido a que la OA como se dijo anteriormente deriva de cierta forma de la OO y gracias a la extensión realizada al metamodelo de UML para que trate la OA, podemos utilizar OCL para definir las métricas necesarias en la OA.

5. EFECTOS DE LA POA EN LAS METRICAS DE CHIDAMBER Y KEMERER METRICS (C&K)

Como se dijo la POA puede verse como una extensión de los sistemas orientados a objetos. Pero en un sistema orientado a aspecto la unidad básica es el aspecto en vez de una clase o procedimiento.
Las métricas utilizadas para la evaluación de sistemas OO son aplicables a los sistemas OA, como es el caso de las métricas C&K definidas por Chidamber y Kemerer [7].

Veremos como estas métricas se ven afectadas al aplicarse a sistemas Orientados a Aspectos.

5.1. PESO DE LOS MÉTODOS POR CLASE (WMC)

Cuento más grande es el número de métodos, el árbol de herencia será mas complicado y es más probable que la clase se vuelva más específica disminuyendo así la potencial re-utilización
Los aspectos pueden ayudar a reducir el número de métodos por clase de la siguiente manera:

- Los aspectos combinan el entrecruzamiento de funcionalidades en unidades modulares y encapsuladas. Sin el diseño orientado a aspectos, entrecruzamiento de funcionalidades estaría desparramado en la clase.
- En algunos casos las subclases deben re definir una función de la clase padre de manera tal que pueda definir el comportamiento de su aspecto.

5.2. PROFUNDIDAD DEL ÁRBOL DE HERENCIA (DIT)

Buenos diseños tienen bajos niveles de herencia ya que cuanto más crece el DIT es más probable que las clases de niveles bajos hereden muchos métodos causando dificultades al momento de definir el comportamiento de una clase.
Las subclases pueden definirse con el propósito de aplicar su propia implementación del comportamiento del aspecto, éstas no existirían en sistemas desarrollados con el paradigma de OA. Esto ayuda a reducir la profundidad del árbol de herencia de una clase.

5.3. NÚMERO DE HIJOS (NOC)

Cuantos más descendientes tenga una clase aumenta la re-utilización pero la abstracción representada por la subclase se diluye. Por esto es beneficioso tratar de mantener el NOC en valores intermedios. La utilización de aspectos ayuda a disminuir el NOC.
En este caso se encuentra la misma justificación que en el caso anterior. Ya que solo las subclases especializadas serían definidas, es decir las subclases que modelaran el comportamiento de los componentes del sistema.

5.4. ACOPLAMIENTO ENTRE OBJETOS (CBO)

Un excesivo acoplamiento entre objetos es dañino para el diseño del módulo y evita el re-uso.
La presencia de aspectos disminuye el acoplamiento entre clases, aunque aumenta el acoplamiento entre clases y aspectos.
Esto se debe a que los aspectos son nuevas entidades de las cuales las clases dependen.
Dado que un diseño puede involucrar acoplamiento entre clases, seria mejor que este se diera entre clases y aspectos.

5.5 RESPUESTA POR CLASE (RFC)

RFC incrementa ante la presencia de aspectos. Esto se debe a que el número de entidades con las que se comunica una clase se incrementa, y las clases tienen que comunicarse con los aspectos. El punto a favor en este caso es que los aspectos pueden diseñarse de manera tal que encapsule la lógica y los objetos con los cuales la clase se comunica en forma modular. De esta manera la complejidad de las pruebas disminuye.

6. METRICAS APLICADAS A LA ORIENTACION A ASPECTOS

Como se dijo en las secciones anteriores la unidad básica de OA son los aspectos por esto surge de forma natural la medición los mismos. Lo que se mostrará en esta sección es la medición de los aspectos intentando encontrar métricas que nos ayuden a conocer la complejidad del sistema que se está desarrollando aplicando esta metodología.
Al interiorizarse en la POA [8][9] encontramos diferentes conceptos relacionados a ésta. Los cuales nos indican las posibles métricas necesarias desde el punto de vista del diseño, para poder obtener una apreciación mas objetiva del diseño del sistema realizado. Entre los conceptos mas relevantes de POA s encuentran:

- **Relación entre Clase y Aspectos:** una clase es una componente cuya funcionalidad puede ser encapsulada. Como la POA separa las clases de los aspectos un aspecto puede relacionarse con una o varias clases. Si existe una relación entre una clase y un aspecto es porque ésta se ve afectada por dicho aspecto.

- **Clase Tejida**: Al utilizar un tejedor de aspectos, el código de las clases y los aspectos se mezclan y se genera como resultado una clase tejida. La estructura de la clase tejida depende del tejedor de aspectos y del lenguaje de programación que se haya utilizado para generarla.

- **Puntos de enlace**: son una clase especial de interfaz entre los aspectos y los módulos del lenguaje de componentes. Son los lugares del código en los que éste se puede aumentar con comportamientos adicionales. Estos comportamientos se especifican en los aspectos.

- **Clases:** unidad básica que encapsula el comportamiento del sistema a través de las distintas funcionalidades y no contiene el comportamiento de los aspectos que la afectan.

Se propone entonces, trabajar con métricas desde el diseño del sistema aplicándolo a UML, para realizar esto; se trabajara con las extensiones realizadas a UML para soportar OA [10][11]. Luego se definirá la semántica de las mismas con OCL. Algunas de las métricas sobre las cuales se trabaja son:

- **Cantidad de Aspectos**: al ser el aspecto la unidad básica de la POA es deseable conocer la cantidad de aspectos que afectan al sistema total o a parte del mismo. Esto nos puede ayudar a justificar el uso de esta metodología ya que si la cantidad de aspectos es muy alta; esta metodología nos beneficia; en cambio si la cantidad es baja nos puede complicar el diseño en vez de mejorarlo.

- **Cantidad de relaciones existentes entre aspectos y una clase**: una clase puede ser atravesada por más de un aspecto, aumentando así la complejidad de la clase.

- **Cantidad de clases relacionadas con un mismo aspecto**: dado que varias clases se pueden relacionar con un mismo aspecto, es importante conocer esta cantidad para conocer así la complejidad del sistema. Ya que el diseño del comportamiento de aspectos puede verse afectado por esto.

- **Cantidad de puntos de enlace en una clase:** al ser los puntos de enlace los lugares en los que se agrega el comportamiento de los aspectos, esta cantidad varia de la cantidad de

aspectos que atraviesan una clase ya que dicha clase puede ser atravesada por un mismo aspecto en diferentes oportunidades.

- **Cantidad de Clases Tejidas:** la generación de la clase tejida depende del lenguaje OA que se utilice. En el caso que se generen clases tejidas el número de éstas me indica la complejidad y reutilización de la clase que intervino para generar la misma.

- **Reutilización de una clase**: se sabe que una clase se puede relacionar con mas de un aspecto, cuanto más se relacione una clase con éstos mas disminuye la reutilización de la misma.

Ahora si se considera el metamodelo de UML extendido para OA, se puede especificar las distintas métricas a través de restricciones utilizando OCL [12]. Antes de definir las métricas con OCL se realizará una descripción de cada una de las métricas de manera tal que el paso siguiente, escribirlas con OCL, sea directo.

Cantidad de Aspectos: Dado que Aspect es una metaclase de UML extendido para OA se puede contar la cantidad de instancias de la misma y así conocer la cantidad de aspectos existentes en el diseño del sistema de software.

Cantidad de relaciones existentes entre aspectos y una clase: Dada una clase específica obtener todas las asociaciones del tipo dependencia existentes, esto se realizara a través de la meta clase Association, y de allí contar aquellas que se correspondan a la metaclase Aspect.

Cantidad de clases relacionadas con un mismo aspecto: Este caso es similar al anterior pero dado un aspecto determinar cuantas clases se relacionan con él.

Cantidad de puntos de enlace en una clase: una clase se puede relacionar con uno o mas aspectos esta relación se muestra a través de relaciones de dependencia. Si una clase está relacionada con un aspecto entonces debe contener al menos un punto de enlace en donde se incorpore el comportamiento del aspecto. Se puede decir que la cantidad de puntos de enlace debería corresponderse a la cantidad de aspectos con los que la clase se relaciona.

Cantidad de Clases Tejidas: una clase Tejida se forma a partir de una Clase y de uno o varios Aspectos, entonces esta cantidad esta directamente relacionada con la cantidad de clases que se relacionan con algún aspecto.
Reutilización de una clase: esta métrica se deriva de la cantidad de aspectos de los cuales depende una clase. Cuanto mayor es la cantidad de aspectos con los que se relaciona una clase menor será la reutilización de la misma.
Como se puede ver la mayoría de las métricas utilizan las metaclases: Aspect, Class, Association; para poder calcularse.
Se muestra como ejemplo la definición de una de las métricas anteriores utilizando OCL.

Cantidad de Aspectos

Context Aspect:
CA:Integer
CA=self.allinstances-> size()

De manera similar se trabaja en la definición de las métricas restantes. Se realizarán recolección de datos a partir de proyectos OA. De esta manera se podrán establecer relaciones entre los datos recolectados y los resultados de la aplicación de las métricas.
Además se modelara los mismos proyectos sin aplicar la POA de esta manera al comparar obtenidos de la aplicación de las métricas a proyectos OA así como a los mismos sin la OA, se verán los beneficios y mejoras en el diseño o el aumento de la complejidad en los mismos.

También se puede encontrar algunas relaciones entre las medidas anteriores como:

- **La cantidad de Puntos de enlace con la reutilización de la clase:** cuanto más es afectada una clase por un aspecto menor es la reutilización de la clase.

- **Cantidad de relaciones existentes entre un aspecto y una clase con la reutilización de la misma**: mayor es la cantidad de aspectos que atraviesan una clase menor será la reutilización de la misma.

- **Cantidad de Clases Tejidas con la cantidad de aspectos**: ya que la clase tejida es generada a partir de una clase y un aspecto puede ser que una misma clase tejida contenga el comportamiento de uno o más aspectos dependiendo de las decisiones de diseño.

7. CONCLUSIONES

La POA es una nueva metodología para el desarrollo de sistemas de software la cual permite un manejo más fácil del seguimiento y mantenimiento del software desarrollado. Como toda metodología debe ser evaluada a través de distintas herramientas para poder así calificar el uso de la misma. Una de las herramientas que permiten medir de manera mas objetiva el diseño realizado mediante la aplicación de distintas técnicas, en particular de los sistemas OA, son las métricas. Esto nos lleva a profundizar el análisis de las métricas para poder aplicarlas al diseño OA y encontrar además de los posibles efectos, nuevas métricas que ayuden a calificar objetivamente un sistema OA.
En este trabajo se muestra tanto el efecto de la aplicación de la OA en las métricas existentes para OO en distintas etapas de desarrollo de sistemas de software como así también las posibles métricas que deberían considerarse para medir de forma mas objetiva el diseño realizado aplicando esta metodología.
Como se dijo anteriormente las métricas proporcionan una vista más objetiva del desarrollo que se realiza, esto es crucial en el momento de estimar recursos ya que no solo cumplir con los requisitos del cliente es importante sino también limitar y estimar los recursos necesarios. Por esto que el análisis de las métricas aplicadas a distintas etapas del desarrollo de software es un tema que debe ser profundizado, en particular par la Orientación a Aspectos ya que su uso en desarrollos grandes aumenta crecientemente.
Es entonces la intención de este trabajo realizar éste análisis para ayudar a los desarrolladores de software que decidan incorporar la OA en su sistema de software.
Se muestra además la extensión de UML para OA y la posible aplicación de OCL en la definición de métricas OA. Este último punto es sobre el cual se trabaja actualmente.

8 FUTUROS TRABAJOS

Sabemos que las métricas nos permiten calificar el diseño de software, pero aún así

la mayor parte del diseño Orientado a Aspectos es subjetivo por lo que la definición formal de estas métricas proporciona una vista mas objetiva de la calidad del diseño. En futuros trabajos se realizara la definición formal de la semántica de la totalidad de las métricas propuestas para sistemas Orientados a Aspectos haciendo uso de OCL aplicado sobre la extensión de UML para OA. Se definirán métricas aplicables tanto a la unidad básica de la OA, los aspectos; así como para diferentes artefactos que intervienen en ella.

9 REFERENCIAS

[1] Zhao J.;Measuring Coupling in Aspect-Oriented Systems; Department of Computer Science and Engineering; Fukuoka Institute of Technology

[2] Quintero A.; Visión General de la Programación Orientada a Aspectos. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla.

[3] Fuentes L., Vallecillo A.; Una Introducción a los Perfiles UML; Revista Novatica 2004.

[4] Booch Grady, Rumbaugh James, Jacobson Ivar, “The Unified Modeling Language User Guide”, Addison Wesley Longman, Inc., 1999.

[5] Object Management Group. UML 2.0 Infrastructure Specification, OMG document ptc/03-09-15, 2003.

[6] Jos Warner; Anneke Kleppe; “The Object Constraint Language” Precise Modeling with UML; Eddison Wesley Publishing 1999

[7] Zakaria; Hosny; Metrics for Aspect-Oriented Software Design. The American University in Cairo.

[8] Nieto Moreno; Introducción a la Programación Orientada a Aspectos, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, España

[9] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier and J.Irwin, Aspect-Oriented Programming, Xerox Palo Alto Research Center, 1997

[10] J. Suzuki, Y. Yamamoto. Extending UML with Aspect: Aspect Support in the Design Phase. 3er Aspect-Oriented Programming (AOP) Workshop at ECOOP’99.

[11] OMG. Unified Modeling Language Specification. v. 1.3, 1ª Edición. Marzo, 2000. http://www.omg.org/technology/documents/formal/unified_modeling_language.htm. UML extendido para OA.

[12] D. Riesco, G. Montejano, R. Uzal, et al, "A Technique Based on the OMG Metamodel: A Definition of Object Oriented Metrics applied to UML Models", The 3rd ACS/IEEE International Conference on Computer Systems and Applications January 3-6, 2005, Cairo, Egypt. www.ieee.org. IEEE Press. Pg. 118. ISBN: 0-7803-8735-X, IEEE Catalog Number: 05EX949, Library of Congress Number: 2004110879